Lab Exercise 5 - Today afternoon

- Room ETH HG E 1.1 from 13:15 to 15:00
- Work description: 8-point algorithm

Estimated poses and 3D structure
2-View Geometry: Recap

- Depth from stereo (i.e., stereo vision)
  - **Assumptions**: K, T and R are known.
  - **Goal**: Recover the 3D structure from images

- 2-view Structure From Motion:
  - **Assumptions**: none (K, T, and R are unknown).
  - **Goal**: Recover simultaneously 3D scene structure, camera poses (up to scale), and intrinsic parameters from two different views of the scene
Outline

• Two-View Structure from Motion
• Robust Structure from Motion
Structure from Motion (SFM)

- **Problem formulation:** Given \( n \) point *correspondences* between two images, \( \{ p^i_1 = (u^i_1, v^i_1), \ p^i_2 = (u^i_2, v^i_2) \} \), simultaneously estimate the 3D points \( P^i \), the camera relative-motion parameters \( (R, T) \), and the camera intrinsics \( K_1, K_2 \) that satisfy:

\[
\begin{bmatrix}
\lambda_1 \begin{bmatrix} u^i_1 \\ v^i_1 \\ 1 \end{bmatrix} = K_1 [T|0] \begin{bmatrix} X^i_w \\ Y^i_w \\ Z^i_w \end{bmatrix} \\
\lambda_2 \begin{bmatrix} u^i_2 \\ v^i_2 \\ 1 \end{bmatrix} = K_2 [R|T] \begin{bmatrix} X^i_w \\ Y^i_w \\ Z^i_w \end{bmatrix}
\end{bmatrix}
\]
Structure from Motion (SFM)

- Two variants exist:
  - **Calibrated camera(s)** \( \Rightarrow K_1, K_2 \) are known
  - **Uncalibrated camera(s)** \( \Rightarrow K_1, K_2 \) are unknown

\[ R, T = ? \]

\[ P^i = ? \]
Structure from Motion (SFM)

- Let’s study the case in which the cameras are calibrated.
- For convenience, let’s use normalized image coordinates.
- Thus, we want to find $R, T, P^i$ that satisfy

$$\begin{bmatrix} \bar{u} \\ \bar{v} \\ 1 \end{bmatrix} = K^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$
Scale Ambiguity

If we rescale the entire scene by a constant factor (i.e., similarity transformation), the projections (in pixels) of the scene points in both images remain exactly the same:
Scale Ambiguity

• In monocular vision, it is therefore **not possible** to recover the absolute scale of the scene!
  • Stereo vision?

• Thus, only **5 degrees of freedom** are measurable:
  • 3 parameters to describe the **rotation**
  • 2 parameters for the **translation up to a scale** (we can only compute the direction of translation but not its length)
Structure From Motion (SFM)

- How many knowns and unknowns?
  - \(4n\) knowns:
    - \(n\) correspondences; each one \((u^i_1, v^i_1)\) and \((u^i_2, v^i_2), i = 1 \ldots n\)
  - \(5 + 3n\) unknowns
    - 5 for the motion up to a scale (rotation-> 3, translation->2)
    - \(3n = \) number of coordinates of the \(n\) 3D points

- Does a solution exist?
  - If and only if the number of independent equations \(\geq\) number of unknowns
    \(\Rightarrow 4n \geq 5 + 3n \Rightarrow n \geq 5\)
  - First analytical solution for 5 points by Kruppa in 1913. The equations yield to a 10 degree order polynomial, which has up to 10 solutions including complex ones.
Cross Product (or Vector Product)

\[ \vec{a} \times \vec{b} = \vec{c} \]

• Vector cross product takes two vectors and returns a third vector that is perpendicular to both inputs, with a direction given by the right-hand rule and a magnitude equal to the area of the parallelogram that the vectors span:

\[ \vec{a} \cdot \vec{c} = 0 \]
\[ \vec{b} \cdot \vec{c} = 0 \]
\[ \|\vec{c}\| = \|\vec{a}\|\|\vec{b}\|\sin(\theta) \]

• So \( \vec{c} \) is perpendicular to both \( \vec{a} \) and \( \vec{b} \) (which means that the dot product is 0)
• Also, recall that the cross product of two parallel vectors is 0
• The cross product between \( \vec{a} \) and \( \vec{b} \) can also be expressed in matrix form as the product between the skew-symmetric matrix of \( \vec{a} \) and a vector \( \vec{b} \)

\[
\begin{bmatrix}
0 & -a_z & a_y \\
a_z & 0 & -a_x \\
-a_y & a_x & 0
\end{bmatrix}
\begin{bmatrix}
b_x \\
b_y \\
b_z
\end{bmatrix}
= [\vec{a}] \times \vec{b}
\]
Epipolar Geometry

\[ \bar{p}_1 = \begin{bmatrix} \bar{u}_1 \\ \bar{v}_1 \\ 1 \end{bmatrix}, \quad \bar{p}_2 = \begin{bmatrix} \bar{u}_2 \\ \bar{v}_2 \\ 1 \end{bmatrix} \]

\( p_1, p_2, T \) are coplanar:

\[ p_2^T \cdot n = 0 \Rightarrow p_2^T \cdot (T \times p_1') = 0 \Rightarrow p_2^T \cdot (T \times (Rp_1)) = 0 \]

\[ \Rightarrow p_2^T [T]_x R \ p_1 = 0 \Rightarrow p_2^T E \ p_1 = 0 \quad \text{epipolar constraint} \]

\[ E = [T]_x R \quad \text{essential matrix} \]
Epipolar Geometry

\[ \bar{p}_1 = \begin{bmatrix} \bar{u}_1 \\ \bar{v}_1 \\ 1 \end{bmatrix} \quad \bar{p}_2 = \begin{bmatrix} \bar{u}_2 \\ \bar{v}_2 \\ 1 \end{bmatrix} \quad \text{Normalized image coordinates} \]

\[ \bar{p}_2^T E \bar{p}_1 = 0 \quad \text{Epipolar constraint or Longuet-Higgins equation (1981)} \]

\[ E = [T] \times R \quad \text{Essential matrix} \]

- The Essential Matrix can be decomposed into \( R \) and \( T \) recalling that \( E = [T] \times R \). Four distinct solutions for \( R \) and \( T \) are possible.

Exercise

- Compute the Essential matrix for the case of two rectified stereo images

Rectified case

\[
R = I_{3 \times 3}
\]

\[
T = \begin{bmatrix}
-b \\
0 \\
0
\end{bmatrix} \rightarrow [T]_x = \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & b \\
0 & -b & 0
\end{bmatrix} \rightarrow E = \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & b \\
0 & -b & 0
\end{bmatrix}
\]
How to compute the Essential Matrix?

- If we don’t know $\mathbf{R}$ and $\mathbf{T}$, can we estimate $\mathbf{E}$ from two images?
- Yes, given at least 5 correspondences
How to compute the Essential Matrix?

- Kruppa showed in 1913 that 5 image correspondences is the minimal case. However, his solution was not efficient.
- In 1996, Philipp proposed an iterative solution
- Only in 2004, the first efficient and non iterative solution was proposed. It uses Groebner basis decomposition [Nister, CVPR’2004]..

- The first popular solution uses 8 points and is called the 8-point algorithm or Longuet-Higgins algorithm (1981). Because of its ease of implementation, it is still used today (e.g., NASA rovers).

The 8-point algorithm

- The Essential matrix $E$ is defined by
  \[ \overline{p}_2^T \ E \overline{p}_1 = 0 \]

- Each pair of point correspondences $\overline{p}_1 = (\overline{u}_1, \overline{v}_1, 1)^T$, $\overline{p}_2 = (\overline{u}_2, \overline{v}_2, 1)$ provides a linear equation:
  \[ \overline{p}_2^T \ E \overline{p}_1 = 0 \]

  \[ E = \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} \]

  \[ \overline{u}_2 \overline{u}_1 e_{11} + \overline{u}_2 \overline{v}_1 e_{12} + \overline{u}_2 e_{13} + \overline{v}_2 \overline{u}_1 e_{21} + \overline{v}_2 \overline{v}_1 e_{22} + \overline{v}_2 e_{23} + \overline{u}_1 e_{31} + \overline{v}_1 e_{32} + e_{33} = 0 \]
The 8-point algorithm

• For $n$ points, we can write

$$
\begin{bmatrix}
\bar{u}_2^{-1} & \bar{u}_1^{-1} & \bar{u}_2^{-1} & \bar{v}_2^{-1} & \bar{v}_1^{-1} & \bar{v}_2^{-1} & \bar{u}_1^{-1} & \bar{v}_1^{-1} \\
\bar{u}_2^{-2} & \bar{u}_1^{-2} & \bar{u}_2^{-2} & \bar{v}_2^{-2} & \bar{v}_1^{-2} & \bar{v}_2^{-2} & \bar{u}_1^{-2} & \bar{v}_1^{-2} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\bar{u}_2^{-n} & \bar{u}_1^{-n} & \bar{u}_2^{-n} & \bar{v}_2^{-n} & \bar{v}_1^{-n} & \bar{v}_2^{-n} & \bar{u}_1^{-n} & \bar{v}_1^{-n}
\end{bmatrix}
\begin{bmatrix}
e_{11} \\
e_{12} \\
e_{13} \\
e_{21} \\
e_{22} \\
e_{23} \\
e_{31} \\
e_{32} \\
e_{33}
\end{bmatrix}
= 0
$$

$Q$ (this matrix is known)

$\bar{E}$ (this matrix is unknown)
The 8-point algorithm

\[ Q \cdot \bar{E} = 0 \]

Minimal solution

- \( Q_{(n \times 9)} \) should have rank 8 to have a unique (up to a scale) non-trivial solution \( \bar{E} \)
- Each point correspondence provides 1 independent equation
- Thus, 8 point correspondences are needed

Over-determined solution

- \( n > 8 \) points
- A solution is to minimize \( ||Q \bar{E}||^2 \) subject to the constraint \( ||\bar{E}||^2 = 1 \).
  The solution is the eigenvector corresponding to the smallest eigenvalue of the matrix \( Q^T Q \) (because it is the unit vector \( x \) that minimizes \( ||Qx||^2 = x^T Q^T Qx \)).
- It can be solved through Singular Value Decomposition (SVD). Matlab instructions:
  - \([U,S,V] = \text{svd}(Q);\)
  - \( Ev = V(:,9); \)
  - \( E = \text{reshape}(Ev,3,3)';\)
8-point algorithm: Matlab code

• A few lines of code. Go to the exercise this afternoon to learn to implement it 😊
8-point algorithm: Matlab code

• function E = calibrated_eightpoint( p1, p2)

• p1 = p1'; % 3xN vector; each column = [u;v;1]
• p2 = p2'; % 3xN vector; each column = [u;v;1]

• Q = [p1(:,1).*p2(:,1) , ...
  p1(:,2).*p2(:,1) , ...
  p1(:,3).*p2(:,1) , ...
  p1(:,1).*p2(:,2) , ...
  p1(:,2).*p2(:,2) , ...
  p1(:,3).*p2(:,2) , ...
  p1(:,1).*p2(:,3) , ...
  p1(:,2).*p2(:,3) , ...
  p1(:,3).*p2(:,3)] ;

• [U,S,V] = svd(Q);
• Eh = V(:,9);

• E = reshape(Eh,3,3)';
Interpretation of the 8-point algorithm

The 8-point algorithm seeks to minimize the following algebraic error

$$\sum_{i=1}^{N} (\bar{p}_{2}^{iT} E \bar{p}_{1}^{i})^2$$

Using the definition of dot product, it can be observed that

$$\bar{p}_{2}^{T} \cdot E \bar{p}_{1} = \|\bar{p}_{2}\|\|E \bar{p}_{1}\|\cos(\theta)$$

We can see that this product depends on the angle $\theta$ between $\bar{p}_{1}$ and the normal $E \bar{p}_{1}$ to the epipolar plane. It is non zero when $\bar{p}_{1}, \bar{p}_{2}$, and $T$ are not coplanar.
Extract R and T from E
(this slide will not be asked at the exam)

• Singular Value Decomposition: \( E = U \sum V^T \)

• Enforcing rank-2 constraint: set smallest singular value of \( \sum \) to 0:

\[
\Sigma = \begin{bmatrix}
\sigma_1 & 0 & 0 \\
0 & \sigma_2 & 0 \\
0 & 0 & 0
\end{bmatrix}
\]

\[
\hat{T} = U \begin{bmatrix}
0 & \pm 1 & 0 \\
\pm 1 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix} \Sigma V^T
\]

\[
\hat{R} = U \begin{bmatrix}
0 & \pm 1 & 0 \\
\pm 1 & 0 & 0 \\
0 & 0 & 1
\end{bmatrix} V^T
\]

\[
\hat{t} = \begin{bmatrix}
0 & -t_z & t_y \\
t_z & 0 & t_x \\
-t_y & t_x & 0
\end{bmatrix} \Rightarrow \hat{t} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}
\]

\[
t = K_2 \hat{t}
\]

\[
R = K_2 \hat{R} K_1^{-1}
\]
Only one solution where points are in front of both cameras

These two views are rotated of 180°
Structure from Motion (SFM)

- Two variants exist:
  - **Calibrated** camera(s) $\Rightarrow K_1, K_2$ are known
    - Uses the Essential Matrix
  - **Uncalibrated** camera(s) $\Rightarrow K_1, K_2$ are unknown
    - Uses the Fundamental Matrix

\[
R, T = ? \\
P^i = ?
\]
The Fundamental Matrix

- Before, we assumed to know the camera intrinsic parameters and we used normalized image coordinates

\[
\begin{bmatrix}
\bar{u}_2^i \\
\bar{v}_2^i \\
1
\end{bmatrix}^T E \begin{bmatrix}
\bar{u}_1^i \\
\bar{v}_1^i \\
1
\end{bmatrix} = 0
\]

\[
\begin{bmatrix}
\bar{u}_1 \\
\bar{v}_1 \\
1
\end{bmatrix} = K_1^{-1} \begin{bmatrix}
u_1^i \\
v_1^i \\
1
\end{bmatrix}
\]

\[
\begin{bmatrix}
\bar{u}_2 \\
\bar{v}_2 \\
1
\end{bmatrix} = K_2^{-1} \begin{bmatrix}
u_2^i \\
v_2^i \\
1
\end{bmatrix}
\]

\[
F = K_2^{-T} E K_1^{-1}
\]

\[
E = [T]_x R
\]

\[
\Rightarrow F = K_2^{-T} [T]_x R K_1^{-1}
\]

Fundamental Matrix
The 8-point Algorithm for the Fundamental Matrix

- The same 8-point algorithm to compute the essential matrix from a set of normalized image coordinates can also be used to determine the Fundamental matrix

\[
\begin{bmatrix}
    u_2^i \\
    v_2^i \\
    1
\end{bmatrix}^T \begin{bmatrix}
    u_1^i \\
    v_1^i \\
    1
\end{bmatrix} = 0
\]
Problem with 8-point algorithm

\[
\begin{bmatrix}
  u_2^1 u_1^1 & u_2^1 v_1^1 & u_2^1 & v_2^1 u_1^1 & v_2^1 v_1^1 & v_2^1 & u_1^1 & v_1^1 & 1 \\
  u_2^2 u_1^2 & u_2^2 v_1^2 & u_2^2 & v_2^2 u_1^2 & v_2^2 v_1^2 & v_2^2 & u_1^2 & v_1^2 & 1 \\
  \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
  u_2^n u_1^n & u_2^n v_1^n & u_2^n & v_2^n u_1^n & v_2^n v_1^n & v_2^n & u_1^n & v_1^n & 1 \\
\end{bmatrix}
\begin{bmatrix}
  f_{11} \\
  f_{12} \\
  f_{13} \\
  f_{21} \\
  f_{22} \\
  f_{23} \\
  f_{31} \\
  f_{32} \\
  f_{33}
\end{bmatrix}
= 0
\]
Problem with 8-point algorithm

- Poor numerical conditioning, which makes results very sensitive to noise

- Can be fixed by rescaling the data: *Normalized 8-point algorithm* [Hartley, 1995]

\[
\begin{bmatrix}
    f_{11} \\
    f_{12} \\
    f_{13} \\
    f_{21} \\
    f_{22} \\
    f_{23} \\
    f_{31} \\
    f_{32} \\
    f_{33}
\end{bmatrix} = 0
\]

<table>
<thead>
<tr>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
<th>Column 4</th>
<th>Column 5</th>
<th>Column 6</th>
<th>Column 7</th>
<th>Column 8</th>
<th>Column 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>~10000</td>
<td>~10000</td>
<td>~100</td>
<td>~10000</td>
<td>~10000</td>
<td>~100</td>
<td>~100</td>
<td>~100</td>
<td>1</td>
</tr>
</tbody>
</table>

![Attention Icon] Orders of magnitude difference between column of data matrix \(\rightarrow\) least-squares yields poor results
Normalized 8-point algorithm (1/3)

- This can be fixed using a normalized 8-point algorithm, which estimates the Fundamental matrix on a set of normalized correspondences (with better numerical properties) and then unnormalizes the result to obtain the fundamental matrix for the given (unnormalized) correspondences.

- Idea: Transform image coordinates so that they are in the range $[-1,1] \times [-1,1]$
- One way is to apply the following rescaling and shift.

\[
\begin{bmatrix}
\frac{2}{700} & 0 & -1 \\
0 & \frac{2}{500} & -1 \\
-1 & 1
\end{bmatrix}
\]
Normalized 8-point algorithm (2/3)

• A more popular way is to rescale the two point sets such that the centroid of each set is 0 and the mean standard deviation $\sqrt{2}$.
• This can be done for every point as follows:

$$\hat{p}^i = \frac{\sqrt{2}}{\sigma} (p^i - \mu)$$

• Where $\mu = \frac{1}{N} \sum_{i=1}^{n} p^i$ is the centroid of the set and $\sigma = \frac{1}{N} \sum_{i=1}^{n} \|p^i - \mu\|^2$ is the mean standard deviation.
• This transformation can be expressed in matrix form using homogeneous coordinates:

$$\begin{bmatrix}
\frac{\sqrt{2}}{\sigma} & 0 & -\frac{\sqrt{2}}{\sigma} \mu^x \\
0 & \frac{\sqrt{2}}{\sigma} & -\frac{\sqrt{2}}{\sigma} \mu^y \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
p^i \\
1
\end{bmatrix}$$
The Normalized 8-point algorithm can be summarized in three steps:

1. Normalize point correspondences: \( \hat{p}_1 = B_1 p_1 \), \( \hat{p}_2 = B_2 p_2 \)
2. Estimate \( \hat{F} \) using normalized coordinates \( \hat{p}_1, \hat{p}_2 \)
3. Compute \( F \) from \( \hat{F} \): \( F = B_2^T \hat{F} B_1 \)

\[
\hat{p}_2^T \hat{F} \hat{p}_1 = 0
\]

\[
\begin{align*}
& B_2^T \hat{p}_2^T \quad \hat{F} \\
\hat{F} = \quad & B_1^T \hat{p}_1^T \\
& B_2^T \hat{F} B_1
\end{align*}
\]
Comparison between Normalized and non-normalized algorithm

<table>
<thead>
<tr>
<th></th>
<th>8-point</th>
<th>Normalized 8-point</th>
<th>Nonlinear refinement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Av. Reprojection error 1</td>
<td>2.33 pixels</td>
<td>0.92 pixel</td>
<td>0.86 pixel</td>
</tr>
<tr>
<td>Av. Reprojection error 2</td>
<td>2.18 pixels</td>
<td>0.85 pixel</td>
<td>0.80 pixel</td>
</tr>
</tbody>
</table>
Error Measures

- The quality of the estimated Fundamental matrix can be measured using different cost functions.
- The first one is the algebraic error that is defined directly in the Epipolar Constraint:

\[
\text{err} = \sum_{i=1}^{N} (\overline{p}_2^T E \overline{p}_1^i)^2
\]

Remember Slide 22 for the geometrical interpretation of this error.
What is the drawback with this error measure?

- This error will exactly be 0 if E is computed from just 8 points (because in this case a solution exists). For more than 8 points, it will not be 0 (due to image noise or outliers (overdetermined system)).
- There are alternative error functions that can be used to measure the quality of the estimated Fundamental matrix: the **Directional Error**, the **Epipolar Line Distance**, or the **Reprojection Error**.
Directional Error

- Sum of the Angular Distances to the Epipolar plane: \( \text{err} = \sum_i (\cos(\theta_i))^2 \)
- From slide 22, we obtain: \( \cos(\theta) = \left( \frac{p_{12}^T \cdot E p_1}{\|p_{12}\| \|E p_1\|} \right)^2 \)
Epipolar Line Distance

- Sum of Squared Epipolar-Line-to-point Distances

\[
err = \sum_{i=1}^{N} d^2(p^i_1, l^i_1) + d^2(p^i_2, l^i_2)
\]

- Cheaper than reprojection error because does not require point triangulation
Reprojection Error

- Sum of the **Squared Reprojection Errors**

\[
err = \sum_{i=1}^{N} \left\| p_1^i - \pi_1(P_i) \right\|^2 + \left\| p_2^i - \pi_2(P_i, R, T) \right\|^2
\]

- Computation is expensive because requires point triangulation
- However it is the most popular because more accurate

How to compute \( P \)? See Slide 28 of past lecture
Outline

• Two-View Structure from Motion
• Robust Structure from Motion
Robust Estimation

- Matched points are usually contaminated by **outliers** (i.e., wrong image matches)
- Causes of outliers are:
  - changes in view point (including scale) and illumination
  - image noise
  - occlusions
  - blur
- For the camera motion to be estimated accurately, outliers must be removed
- This is the task of **Robust Estimation**
Robust Estimation

- Matched points are usually contaminated by outliers (i.e., wrong image matches).
- Causes of outliers are:
  - changes in view point (including scale) and illumination
  - image noise
  - occlusions
  - blur
- For the camera motion to be estimated accurately, outliers must be removed.
- This is the task of Robust Estimation.
Influence of Outliers on Motion Estimation

- Error at the loop closure: 6.5 m
- Error in orientation: 5 deg
- Trajectory length: 400 m

Outliers can be removed using RANSAC [Fishler & Bolles, 1981]
RANSAC (RAndom SAmple Consensus)

- RANSAC is the **standard method for model fitting in the presence of outliers** (very noisy points or wrong data)
- It can be applied to all sorts of problems where the goal is to **estimate the parameters of a model from the data** (e.g., camera calibration, Structure from Motion, DLT, PnP, P3P, Homography, etc.)
- Let’s review RANSAC for line fitting and see how we can use it to do Structure from Motion

RANSAC
RANSAC

- Select sample of 2 points at random
RANSAC

- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
RANSAC

- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- Select data that supports current hypothesis
RANSAC

• Select sample of 2 points at random

• Calculate model parameters that fit the data in the sample

• Calculate error function for each data point

• Select data that supports current hypothesis

• Repeat
RANSAC

- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- Select data that supports current hypothesis
- Repeat
Select the set with the maximum number of inliers obtained within $k$ iterations
RANSAC

How many iterations does RANSAC need?

• Ideally: check all possible combinations of 2 points in a dataset of \( N \) points.

• Number of all pairwise combinations: \( N(N-1)/2 \)
  ⇒ computationally unfeasible if \( N \) is too large.

  example: 1000 points ⇒ need to check all 1000*999/2 ≅ 500’000 possibilities!

• Do we really need to check all possibilities or can we stop RANSAC after some iterations?
  Checking a subset of combinations is enough if we have a rough estimate of the percentage of inliers in our dataset

• This can be done in a probabilistic way
RANSAC

How many iterations does RANSAC need?

• \( w := \text{number of inliers}/N \)
  \( N := \text{total number of data points} \)
  ⇒ \( w \) : fraction of inliers in the dataset ⇒ \( w = P(\text{selecting an inlier-point out of the dataset}) \)

• Assumption: the 2 points necessary to estimate a line are selected independently
  ⇒ \( w^2 = P(\text{both selected points are inliers}) \)
  ⇒ \( 1 - w^2 = P(\text{at least one of these two points is an outlier}) \)

• Let \( k := \text{no. RANSAC iterations executed so far} \)
  ⇒ \((1 - w^2)^k = P(\text{RANSAC never selected two points that are both inliers})\)

• Let \( p := P(\text{probability of success}) \)
  ⇒ \( 1 - p = (1 - w^2)^k \) and therefore:

\[
k = \frac{\log(1 - p)}{\log(1 - w^2)}
\]
How many iterations does RANSAC need?

• The number of iterations $k$ is

$$k = \frac{\log(1 - p)}{\log(1 - w^2)}$$

• Knowing the fraction of inliers $w$, after $k$ RANSAC iterations we will have a probability $p$ of finding a set of points free of outliers.

• Example: if we want a probability of success $p=99\%$ and we know that $w=50\% \Rightarrow k=16$ iterations – these are dramatically fewer than the number of all possible combinations! As you can see, the number of points does not influence the estimated number of iterations, only $w$ does!

• In practice we only need a rough estimate of $w$. More advanced variants of RANSAC estimate the fraction of inliers and adaptively update it at every iteration (how?)
RANSAC applied to Line Fitting

1. Initial: let $A$ be a set of $N$ points
2. repeat
3. Randomly select a sample of 2 points from $A$
4. Fit a line through the 2 points
5. Compute the distances of all other points to this line
6. Construct the inlier set (i.e. count the number of points whose distance $< d$)
7. Store these inliers
8. until maximum number of iterations $k$ reached
9. The set with the maximum number of inliers is chosen as a solution to the problem
RANSAC applied to general model fitting

1. Initial: let $A$ be a set of $N$ points
2. repeat
3. Randomly select a sample of $s$ points from $A$
4. Fit a model from the $s$ points
5. Compute the distances of all other points from this model
6. Construct the inlier set (i.e. count the number of points whose distance $< d$)
7. Store these inliers
8. until maximum number of iterations $k$ reached
9. The set with the maximum number of inliers is chosen as a solution to the problem

$$k = \frac{\log(1 - p)}{\log(1 - w^s)}$$
The Three Key Ingredients of RANSAC

In order to implement RANSAC for Structure From Motion (SFM), we need three key ingredients:

1. What’s the model in SFM?
2. What’s the minimum number of points to estimate the model?
3. How do we compute the distance of a point from the model? In other words, can we define a distance metric that measures how well a point fits the model?
Answers

1. **What’s the model** in SFM?
   - The **Essential Matrix** (for calibrated cameras) or the **Fundamental Matrix** (for uncalibrated cameras)
   - Alternatively, $R$ and $T$

2. **What’s the minimum number of points** to estimate the model?
   1. We know that 5 points is the theoretical minimum number of points
   2. However, if we use the 8-point algorithm, then 8 is the minimum

3. **How do we compute the distance** of a point from the model?
   1. We can use the epipolar constraint $(\tilde{p}_2^T E \tilde{p}_1 = 0$ or $p_2^T F p_1 = 0)$ to measure how well a point correspondence verifies the model $E$ or $F$, respectively. However, the **Directional error**, the **Epipolar line distance**, or the **Reprojection error (even better)** are used (we already saw why)
Example: 8-point RANSAC applied to SfM

- Let’s consider the following image pair and its image correspondences (e.g., Harris, SIFT, etc.), denoted by arrows
Example: 8-point RANSAC applied to SfM

- Let’s consider the following image pair and its image correspondences (e.g., Harris, SIFT, etc.), denoted by arrows.
- For convenience, we overlay the features of the second image in the first image and use arrows to denote the motion vectors of the features.
Example: 8-point RANSAC applied to SfM

• Let’s consider the following image pair and its image correspondences (e.g., Harris, SIFT, etc.), denoted by arrows
• For convenience, we overlay the features of the second image in the first image and use arrows to denote the motion vectors of the features

1. Randomly select 8 point correspondences
Example: 8-point RANSAC applied to SfM

- Let’s consider the following image pair and its image correspondences (e.g., Harris, SIFT, etc.), denoted by arrows.
- For convenience, we overlay the features of the second image in the first image and use arrows to denote the motion vectors of the features.

1. Randomly select 8 point correspondences
2. Fit the model to all other points and count the inliers.
Example: 8-point RANSAC applied to SfM

- Let’s consider the following image pair and its image correspondences (e.g., Harris, SIFT, etc.), denoted by arrows.
- For convenience, we overlay the features of the second image in the first image and use arrows to denote the motion vectors of the features.

1. Randomly select 8 point correspondences.
2. Fit the model to all other points and count the inliers.
3. Repeat from 1.
Example: 8-point RANSAC applied to SfM

• Let’s consider the following image pair and its image correspondences (e.g., Harris, SIFT, etc.), denoted by arrows.

• For convenience, we overlay the features of the second image in the first image and use arrows to denote the motion vectors of the features.
Example: 8-point RANSAC applied to SfM

- Let’s consider the following image pair and its image correspondences (e.g., Harris, SIFT, etc.), denoted by arrows.
- For convenience, we overlay the features of the second image in the first image and use arrows to denote the motion vectors of the features.

1. Randomly select 8 point correspondences
Example: 8-point RANSAC applied to SfM

- Let’s consider the following image pair and its image correspondences (e.g., Harris, SIFT, etc.), denoted by arrows.
- For convenience, we overlay the features of the second image in the first image and use arrows to denote the motion vectors of the features.

1. Randomly select 8 point correspondences.
2. Fit the model to all other points and count the inliers.

Image 1
Example: 8-point RANSAC applied to SfM

- Let’s consider the following image pair and its image correspondences (e.g., Harris, SIFT, etc.), denoted by arrows
- For convenience, we overlay the features of the second image in the first image and use arrows to denote the motion vectors of the features

1. Randomly select 8 point correspondences
2. Fit the model to all other points and count the inliers
3. Repeat from 1 for $k$ times

$$k = \frac{\log(1 - p)}{\log(1 - (1 - \varepsilon)^8)}$$
RANSAC iterations $k$ vs. $s$

$k$ is exponential in the number of points $s$ necessary to estimate the model:

- **8-point RANSAC**
  - Assuming
    - $p = 99\%$,
    - $\varepsilon = 50\%$ (fraction of outliers)
    - $s = 8$ points (8-point algorithm)

  $$k = \frac{\log(1 - p)}{\log(1 - (1 - \varepsilon)^s)} = 1177 \text{ iterations}$$

- **5-point RANSAC**
  - Assuming
    - $p = 99\%$,
    - $\varepsilon = 50\%$ (fraction of outliers)
    - $s = 5$ points (5-point algorithm of David Nister (2004))

  $$k = \frac{\log(1 - p)}{\log(1 - (1 - \varepsilon)^s)} = 145 \text{ iterations}$$

- **2-point RANSAC (e.g., line fitting)**
  - Assuming
    - $p = 99\%$,
    - $\varepsilon = 50\%$ (fraction of outliers)
    - $s = 2$ points

  $$k = \frac{\log(1 - p)}{\log(1 - (1 - \varepsilon)^s)} = 16 \text{ iterations}$$
RANSAC iterations $k$ vs. $\varepsilon$

- $k$ is increases exponentially with the fraction of outliers $\varepsilon$
RANSAC iterations

- As observed, $k$ is exponential in the number of points $s$ necessary to estimate the model.
- The 8-point algorithm is extremely simple and was very successful; however, it requires more than 1177 iterations.
- Because of this, there has been a large interest by the research community in using smaller motion parameterizations (i.e., smaller $s$).
- The first efficient solution to the minimal-case solution (5-point algorithm) took almost a century (Kruppa 1913 $\rightarrow$ Nister 2004).
- The 5-point RANSAC (Nister 2004) only requires 145 iterations; however:
  - The 5-point algorithm can return up to 10 solutions of $E$ (worst case scenario).
  - The 8-point algorithm only returns a unique solution of $E$.

Can we use less than 5 points?
Yes, if you use motion constraints!
Planar Motion

Planar motion is described by three parameters: $\theta$, $\varphi$, $\rho$

\[
R = \begin{bmatrix}
\cos \theta & -\sin \theta & 0 \\
\sin \theta & \cos \theta & 0 \\
0 & 0 & 1 \\
\end{bmatrix} \quad T = \begin{bmatrix}
\rho \cos \varphi \\
\rho \sin \varphi \\
0 \\
\end{bmatrix}
\]

Let’s compute the Epipolar Geometry

\[
E = [T]_x R \quad \text{Essential matrix}
\]

\[
\overrightarrow{p_2}^T E \overrightarrow{p_1} = 0 \quad \text{Epipolar constraint}
\]
Planar Motion

Planar motion is described by three parameters: $\theta$, $\varphi$, $\rho$

$$R = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad T = \begin{bmatrix} \rho \cos \varphi \\ \rho \sin \varphi \\ 0 \end{bmatrix}$$

Let's compute the Epipolar Geometry

$$[T]_x = \begin{bmatrix} 0 & 0 & \rho \sin \varphi \\ 0 & 0 & -\rho \cos \varphi \\ -\rho \sin \varphi & \rho \cos \varphi & 0 \end{bmatrix}$$

$$E = [T]_x R = \begin{bmatrix} 0 & 0 & \rho \sin \varphi \\ 0 & 0 & -\rho \cos \varphi \\ -\rho \sin \varphi & \rho \cos \varphi & 0 \end{bmatrix} \cdot \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
Planar Motion

Planar motion is described by three parameters: $\vartheta$, $\varphi$, $\rho$

\[ R = \begin{bmatrix} \cos \vartheta & -\sin \vartheta & 0 \\
\sin \vartheta & \cos \vartheta & 0 \\
0 & 0 & 1 \end{bmatrix} \quad T = \begin{bmatrix} \rho \cos \varphi \\
\rho \sin \varphi \\
0 \end{bmatrix} \]

Let's compute the Epipolar Geometry

\[ [T]_x = \begin{bmatrix} 0 & 0 & \rho \sin \varphi \\
0 & 0 & -\rho \cos \varphi \\
-\rho \sin \varphi & \rho \cos \varphi & 0 \end{bmatrix} \]

\[ E = [T]_x R = \begin{bmatrix} 0 & 0 & \rho \sin (\varphi) \\
0 & 0 & -\rho \cos (\varphi) \\
-\rho \sin (\varphi - \theta) & \rho \cos (\varphi - \theta) & 0 \end{bmatrix} \]
Planar Motion

Planar motion is described by three parameters: $\theta$, $\phi$, $\rho$

$$
R = \begin{bmatrix}
\cos \theta & -\sin \theta & 0 \\
\sin \theta & \cos \theta & 0 \\
0 & 0 & 1
\end{bmatrix} \quad T = \begin{bmatrix}
\rho \cos \phi \\
\rho \sin \phi \\
0
\end{bmatrix}
$$

Observe that $E$ has 2DoF ($\theta$, $\phi$, because $\rho$ is the scale factor); thus, 2 correspondences are sufficient to estimate $\theta$ and $\phi$ [“2-Point RANSAC”, Ortin, 2001]

$$
E = [T] x R = 
\begin{bmatrix}
0 & 0 & \rho \sin (\phi) \\
0 & 0 & -\rho \cos (\phi) \\
-\rho \sin (\phi - \theta) & \rho \cos (\phi - \theta) & 0
\end{bmatrix}
$$
Can we use less than 2 point correspondences?

Yes, if we exploit wheeled vehicles with non-holonomic constraints
Planar & Circular Motion (e.g., cars)

Wheeled vehicles, like cars, follow locally-planar circular motion about the Instantaneous Center of Rotation (ICR).

Example of Ackerman steering principle

Locally-planar circular motion
Planar & Circular Motion (e.g., cars)

Wheeled vehicles, like cars, follow locally-planar circular motion about the Instantaneous Center of Rotation (ICR)

![Diagram of Ackerman steering principle](image1)

![Diagram of locally-planar circular motion](image2)

Example of Ackerman steering principle

Locally-planar circular motion

\[ \phi = \theta / 2 \Rightarrow \text{only 1 DoF } (\theta); \]

thus, only 1 point correspondence is needed

This is the smallest parameterization possible and results in
the most efficient algorithm for removing outliers

Let’s compute the Epipolar Geometry

\[ E = [T]_x R \quad \text{Essential matrix} \]

\[ \bar{p}_2^T E \bar{p}_1 = 0 \quad \text{Epipolar constraint} \]
Planar & Circular Motion (e.g., cars)

\[
R = \begin{bmatrix}
\cos \theta & -\sin \theta & 0 \\
\sin \theta & \cos \theta & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
T = \begin{bmatrix}
\rho \cos \frac{\theta}{2} \\
\rho \sin \frac{\theta}{2} \\
0
\end{bmatrix}
\]

Let’s compute the Epipolar Geometry

\[
E = [T]_x R =
\begin{bmatrix}
0 & 0 & \rho \sin \frac{\theta}{2} \\
0 & 0 & -\rho \cos \frac{\theta}{2} \\
-\rho \sin \frac{\theta}{2} & \rho \cos \frac{\theta}{2} & 0
\end{bmatrix}
\cdot
\begin{bmatrix}
\cos \theta & -\sin \theta & 0 \\
\sin \theta & \cos \theta & 0 \\
0 & 0 & 1
\end{bmatrix} =
\begin{bmatrix}
0 & 0 & \rho \sin \frac{\theta}{2} \\
0 & 0 & \rho \cos \frac{\theta}{2} \\
\rho \sin \frac{\theta}{2} & -\rho \cos \frac{\theta}{2} & 0
\end{bmatrix}
\]

\[\phi = \theta/2\]
Planar & Circular Motion (e.g., cars)

\[
R = \begin{bmatrix}
\cos \theta & -\sin \theta & 0 \\
\sin \theta & \cos \theta & 0 \\
0 & 0 & 1
\end{bmatrix}
\quad
T = \begin{bmatrix}
\rho \cos \frac{\theta}{2} \\
\rho \sin \frac{\theta}{2} \\
0
\end{bmatrix}
\]

Let’s compute the Epipolar Geometry

\[
p_2^T E p_1 = 0 \quad \Rightarrow \quad \sin \left( \frac{\theta}{2} \right) \cdot (u_2 + u_1) + \cos \left( \frac{\theta}{2} \right) \cdot (v_2 - v_1) = 0
\]

\[
\theta = -2 \tan^{-1} \left( \frac{v_2 - v_1}{u_2 + u_1} \right)
\]
1-Point RANSAC algorithm

Only 1 iteration!
The most efficient algorithm for removing outliers, up to 1000 Hz

1-Point RANSAC is ONLY used to find the inliers.
Motion is then estimated from them in 6DOF

Compute $\theta$ for every point correspondence

$$\theta = -2 \tan^{-1} \left( \frac{v_2 - v_1}{u_2 + u_1} \right)$$
Comparison of RANSAC algorithms

\[ N = \frac{\log(1 - p)}{\log(1 - (1 - \varepsilon)^s)} \]

where we typically use \( p = 99\% \)

<table>
<thead>
<tr>
<th></th>
<th>8-Point RANSAC</th>
<th>5-Point RANSAC</th>
<th>2-Point RANSAC</th>
<th>1-Point RANSAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numb. of</td>
<td>&gt; 1177</td>
<td>&gt; 145</td>
<td>&gt; 16</td>
<td>= 1</td>
</tr>
<tr>
<td>iterations</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Visual Odometry with 1-Point RANSAC

Work in different environments

Urban

Things to remember

• SFM from 2 view
  – Calibrated and uncalibrated case
  – Proof of Epipolar Constraint
  – 8-point algorithm and algebraic error
  – Normalized 8-point algorithm
  – Algebraic, directional, Epipolar line distance, Reprojection error
  – RANSAC and its application to SFM
  – 8 vs 5 vs 1 point RANSAC, pros and cons

• Readings:
  – Ch. 14.2 of Corke book
  – CH. 7.2 of Szeliski book